Protecting Sensitive Data in Web Browsers with ScriptPolice

Brad Karp UCL with Petr Marchenko (UCL) and Úlfar Erlingsson (Google)



HCSS 2013 May 7th, 2013

Should you trust your browser?

- Browsers handle sensitive data
 - e.g., email, online banking, medical records
- Browser executes untrusted JavaScript written by multiple parties
 - Pages: code written by site operators
 - Extensions: code written by third-party developers
- Potential for attacks on confidentiality
 - Pages exploit extensions
 - Extensions leak users' sensitive data from pages
- This talk:
 - These attacks are real
 - How to defend against them

Browser Primer: Extensions







Browser Primer: SOP and Extension Implementation



Browser Primer: SOP and Extension Implementation



Problem: Malicious Extensions







10m users

Problem: Malicious Extensions



00	Step 2: Authenticate - Log in - myBarclay:	5			_
Barclays Bank Plc [GB] https://bank.barc	lays.co.uk/olb/auth/LoginLink.action				1
	Moore site About bar	clays Contact us	security Legi	ai Accessionity	Home
1. Your details / Authenticate				6155.	
Step 2: Authenticate			· Lucios		
PINsentry user? How to set up You can set up a passcode and men without PINsentry. To set them up the top menu bar. Then look for th You'll still need to use PINsentry to time.	p passcode access morable word so you can log in to Online Banking jog in with Philosoftry and go to 'Account services' in e link under 'Customise my ste'. make payments to people or companies for the first				
 Log in using the PINsentry card reader with debit, cash or Authentication card 	Cog in using Mobile PINsentry from your Barclays Mobile Banking app				
ast 4 digits of your card number 😧	Yuu can use any of your Chip & PIN enabled Barclays debit, cash or authentication cards. What if I don't have the right type of card? ●				
Nobile PINsentry If you have downloaded the latest version	of Barclays Mobile Banking you can also				*



10m users

Problem: Malicious Pages





Problem: Malicious Pages



We ve just released a new WordPress theme at Theme Trust called Ink. The theme is simple, clean, elegant and responsive, allowing your portfolio to adapt to any screen resolution. Check out all the great features this theme has to offer and be ready to give your portfolio a new look.

35816 readers

92k Followers

Del b

Problem: Malicious Pages



ys.co.uk/olb/auth/LoginLink.action						
Mobile site	About Barclays	Contact us	Security	Legal	Accessibility	Home
			· EARCLAYS			10
passcode access						
prable word so you can log in to Online Banking og in with PINsentry and go to 'Account service link under 'Customise my site'.) es' in		74	-	-	
nake payments to people or companies for the	first		- (9		31
C Log in using Mobile PINsentry your Barclays Mobile Banking	ry from g app					1
You can use any of your Chip & PIN er Barclays debit, cash or authentication	nabled cards.					
	Poore site passcode access rable word so you can log in to Online Banking in with PiNaentry and go to 'Account servic in under 'Quatomise my site'. alaie payments to upople or companies for the wour Bardays Mobile PiNsent your Bardays Mobile Bankin	Mobile site About Bardays passcode access Image: State of the st	About Bardays Contact us passcode access Image: State of the sta	Mobile site About Bardays Contact us Security passcode access Image: Security and go to Account services in Image and the topole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security pass payments to poole or companies for the first Image: Security	Notifie site About Bardays Contact us Security Legal	Nobile site Nobile State Contact us Security Legal Accessibility



Are Extensions Vulnerable?

- We've found zero-day vulnerabilities in four popular Chrome extensions...
- ...and designed malicious pages exploiting these extensions...



















Are Extensions Vulnerable?

- We've found zero-day vulnerabilities in four popular Chrome extensions...
- ...and designed malicious pages exploiting these extensions...
- Carlini et al. studied 100 Chrome extensions; found 70 vulnerabilities in 40 of them

N. Carlini, A. P. Felt, and D. Wagner. An Evaluation of the Google Chrome Extension Security Architecture, USENIX Security 2012

Our Solution: ScriptPolice

- A policy system for JavaScript execution in web browsers
- Policies block exfiltration of sensitive data
- Policies are simple and general:
 - A few simple policies "baked into" browser
 - These few policies compatible with wide range of today's extensions and pages
- Full working prototype for V8 JIT-compiled JavaScript engine in Google Chrome browser
- Performance overhead virtually imperceptible to users

ScriptPolice: System Overview

- Browser ships with a few standard policies; confines extension execution with them
 - Prevention policies block pages from injecting scripts into vulnerable extensions; implemented with information flow control (IFC)
 - Containment policies block extensions from exfiltrating sensitive data to network; implemented with discretionary access control (DAC)
- Page developers annotate sensitive page elements (e.g., bank balance, medical diagnostic test name and results)
- Extension developers declare privileges required by an extension with enhanced extension manifest

ScriptPolice: System Overview

- Browser ships with a few standard policies; confines extension execution with them
 - Prevention policies block pages from injecting scripts into vulnerable extensions; implemented with information flow control (IFC)

Two key technical contributions: **Run-time code specialization** of IFC in V8 JIT compiler yields high performance **Discretionary access control at sinks** yields broad policy applicability

by an extension with enhanced extension manifest

ScriptPolice: Interposition



policy decisions: allow, block, throw exception

Prevention Policy

- Prevents pages from injecting code into vulnerable extensions
- Implemented with objectgranularity IFC for JavaScript
 - Label all page data inbound to extension's V8 environment as <page-origin>
 - Propagate labels during JavaScript execution
 - Throw exception upon execution of code labeled <page-origin>
- IFC for preventing script injection not new [Djeric and Goel 2010]
- Our contributions:
 - **JIT-compiled** IFC for JavaScript
 - Faster IFC through run-time specialization



 Previous dynamic, fine-grained IFC systems (e.g., taint tracking) emit label propagation code for every operation

$$r = a1 + a2;$$

 Previous dynamic, fine-grained IFC systems (e.g., taint tracking) emit label propagation code for every operation

```
if (IsLabeled(a1) ||
	IsLabeled(a2))
	Set(label_flag);
r = a1 + a2;
if (IsSet(label_flag))
	Label(r, Labels(a1, a2));
```

Specialize for nonlabeled operations first (no label propagation code)

r = a1 + a2;

Generalize for labeled and non-labeled code later if required

if (IsLabeled(a1) ||
 IsLabeled(a2))
 Set(label_flag);

```
r = a1 + a2;
```

```
if (IsSet(label_flag))
  Label(r, Labels(a1, a2));
```

Specialize for nonlabeled operations first (no label propagation code)

r = a1 + a2;

Generalize for labeled and non-labeled code later if required

if (IsLabeled(a1) ||
 IsLabeled(a2))
 Set(label_flag);

$$r = a1 + a2;$$

Only pay overhead when processing labeled data; non-labeled execution at full speed!

Containment Policy

- Defends pages against malicious extensions
- Blocks exfiltration of sensitive information via network
- Implemented with discretionary access control (DAC) for V8 JavaScript environment
 - Key idea: as needed, apply DAC at the data sink (browser-network boundary) or at the data source (page)
 - Benefits: no implicit flows, generality, simplicity
- Our contributions:
 - DAC policies widely compatible with canonical extension behaviors
 - Automatic policy selection for extensions based on manifest



Challenge: DAC for Legacy Extensions

- Naïve approach: deny extension access to all DOM elements marked sensitive in page
 - Definitely prevents exfiltration of sensitive data by extension—never sees such data
 - But breaks many extensions that must compute over sensitive data to do their job, and never even try to exfiltrate sensitive data!
 - Subtle conflict of interest: page author can mark data "sensitive" to deny extension access! (e.g., ads invulnerable to AdBlock...)
- Insight: need more flexibility than sourcebased DAC provides

Canonical Extension Behaviors

- Local behavior: read from page, process locally, display result; no network communication (e.g., FlashBlock)
- Remote behavior: read from page, send to remote server for processing, display result (e.g., Google Dictionary)
- Promiscuous behavior: read from page, send to remote server unknown at extension installation time, display result (e.g., Download Master)

Canonical Extension Behaviors

 Local behavior: read from page, process locally, display result; no network communication (e.g., FlashBlock)

Observations and a caveat:

Local extensions don't need network access Remote ones do, so risk exfiltrating sensitive data from pages

Covert channels excluded from threat model for now

(e.g., Download Master)

Flexible Containment: DAC Sink and Source



Evaluation

- Implemented ScriptPolice for V8 JIT-compiled JavaScript environment in Google Chrome browser
- Evaluated with dozens of extensions and Alexa top-100 web pages
- Metrics:
 - Browser performance: users essentially unwilling to "pay" much for improved confidentiality
 - Policy compatibility with legacy extensions (i.e., don't break them)
 - Policy efficacy at preventing {injection, exfiltration}

ScriptPolice Is Fast: YouTube Page Load Latency



- Page load latencies are for 25th %ile, median, and 75th %ile over 100 trials
- For most extensions and pages, ScriptPolice increases page load times over baseline by less than 5%, on the order of tens of ms
- Results for other pages and extensions broadly similar

Next Step: Principled Whole-Browser Security with IFC

- Ad hoc, porous implementation of SOP is root of browser vulnerability misery: XSS, CSRF, third-party image/CSS leaks, &c., &c., &c.
- Observation: the SOP is a non-interference IFC policy, but patchily implemented
- Our ongoing work: replace the SOP with pervasive, browser-wide IFC, including exposure of labels to JavaScript code:
 - Stronger isolation than ad hoc SOP
 - More flexible than SOP for, e.g., mashup creation

Next Step: Principled Whole-Browser Security with IFC

• Ad hoc, porous implementation of SOP is root of browser vulnerability misery: XSS, CSRF,

In discussions with Google Chrome and Mozilla Firefox developers about adoption To learn more, read our HotOS 2013 paper (appearing next week!), available at: http://www.cs.ucl.ac.uk/staff/B.Karp/

exposure of labels to JavaScript code:

- Stronger isolation than ad hoc SOP
- More flexible than SOP for, e.g., mashup creation

ScriptPolice: Summary

- Today's browsers don't protect sensitive data robustly, because SOP doesn't (and cannot) constrain extensions
- Prevention and containment: two general policies that protect sensitive data in browsers
- ScriptPolice: practical policy system for V8 JavaScript engine in Chrome browser [code release imminent!]
 - General: supports Containment and Prevention policies for wide range of extensions and pages
 - Fast: native-code IFC; negligible overhead per page load
- Key new techniques:
 - Tailoring DAC Source/Sink to canonical extension behaviors
 - Run-time specialization for fast JIT-compiled IFC